OpenCat: Improving Interoperability of ADS Testing

Qurban Ali
University of Milano-Bicocca
Milan, Italy
q.ali@campus.unimib.it

Andrea Stocco
Technical University of Munich, fortiss GmbH
Munich, Germany
andrea.stocco@tum.de, stocco@fortiss.org

Leonardo Mariani
University of Milano-Bicocca
Milan, Italy
leonardo.mariani@unimib.it

Oliviero Riganelli
University of Milano-Bicocca
Milan, Italy
oliviero.riganelli@unimib.it

Abstract—Testing Advanced Driving Assistance Systems (ADAS), such as lane-keeping functions, requires creating road topologies or using predefined benchmarks. However, the test cases in existing ADAS benchmarks are often designed in specific formats (e.g., OpenDRIVE) and tailored to specific ADAS models. This limits their reusability and interoperability with other simulators and models, making it challenging to assess ADAS functionalities independently of the platform-specific details used to create the test cases.

This paper evaluates the interoperability of SENSODAT, a benchmark developed for ADAS regression testing. We introduce OPENCAT, a converter that transforms OpenDRIVE test cases into the Catmull-Rom spline format, which is widely supported by many current test generators. By applying OPENCAT to the SENSODAT dataset, we achieved high accuracy in converting test cases into reusable road scenarios. To validate the converted scenarios, we used them to evaluate a lane-keeping ADAS model using the Udacity simulator. Both the simulator and the ADAS model operate independently of the technologies underlying SENSODAT, ensuring an unbiased evaluation of the original test cases. Our findings reveal that benchmarks built with specific ADAS models hinder their effective usage for regression testing. We conclude by offering insights and recommendations to enhance the reusability and transferability of ADAS benchmarks for more extensive applications.

Index Terms—*OpenDRIVE, Catmull-Rom Spline, Conversion, Benchmark, Autonomous Driving.*

I. INTRODUCTION

Autonomous Driving Systems (ADS) integrate advanced technologies like adaptive cruise control, parking assistance, and autopilots into a cohesive functional unit. Modern ADS are designed with enhanced capabilities to operate independently with minimal or no human intervention, following a perception-plan-execution framework [1], [2]. The perception component is delegated to Advanced Driver Assistance Systems (ADAS) operated by sophisticated perception systems, including cameras, LiDAR, and various sensors, to interpret complex and dynamic driving environments. These perception systems, in turn, leverage deep neural networks (DNNs) to process sensor data and perform tasks such as object detection, image classification, semantic segmentation, and regression, thereby enabling real-time driving functionalities [1]–[4].

Validating autonomous driving systems across diverse scenarios requires designing extensive testing and simulation environments [5], such as the creation of detailed road topologies, to test the limitations of lane-keeping components [6]–[13], among others. Testing ADAS with challenging road topologies in virtual simulation environments is crucial, not only to drastically reduce the cost of testing autonomous vehicles but also to provide safety guarantees before the software is deployed in production on physical vehicles [10], [14]–[16].

Recent years have seen a growing focus on testing methodologies for ADAS/ADS, particularly concerning test generation techniques [17]–[32]. Other studies have proposed benchmarks for ADAS testing, such as SENSODAT [33], DEEPSCENARIO [34], and SCTRANS [35]. These benchmarks include hundreds of scenarios designed for ADAS regression testing, intended to provide reusable specifications for testing. However, instead of defining the test objectives in a general and reusable manner, they often focus on technical implementations that detail specific test executions for particular simulation platforms and ADAS models. These implementations include both successful and failing test cases, which may not adequately represent corner cases across different ADAS models or simulators [36], [37]. Due to the tight coupling between test implementations, simulation platforms, and ADAS models, we hypothesize that this technical dependency limits the broader applicability and reuse of these benchmarks.

In this paper, we evaluate our preliminary hypothesis by examining the interoperability of high-level scenarios, with a specific focus on SENSODAT, and investigate their potential to provide truly reusable specifications. To achieve this, we adapt SENSODAT scenarios for use in a different simulator with a distinct road format, testing their compatibility with an independent, learning-based ADAS model.

First, we describe OPENCAT, a tool designed to convert the OpenDRIVE road format (used by SENSODAT and

BEAMNG.TECH [38]) into the Catmull-Rom spline representation (utilized by the Udacity simulator [39]). Both formats are integral to the field of ADS/ADAS development. Open-DRIVE [40] is widely adopted in the industry for defining road networks and is commonly used to test Level 4 ADS in complex urban environments. In contrast, Catmull-Rom splines [41] offer a smooth and mathematically precise representation of roads, making them ideal for generating controllable road scenarios to test Level 2 ADAS systems, such as trajectory planning or lane-keeping assistance.

While OPENCAT provides precise conversion from Open-DRIVE data to Catmull-Rom splines (100% accuracy), our second contribution focuses on evaluating the interoperability of the SENSODAT benchmark. Re-simulation experiments conducted in the Udacity self-driving car simulator reveal significant percentage differences in the test pass ratio, ranging from 25% to 50%, emphasizing that benchmarks closely tailored to specific ADAS models may constrain their broader applicability for regression testing. Our paper makes the following contributions:

1) To the best of our knowledge, we present the first openly accessible converter OPENCAT, that transforms the OpenDRIVE road format to a Catmull-Rom spline representation.
2) We show the utility of OPENCAT by applying it to the publicly available SENSODAT [33] dataset, successfully converting 32,580 roads with high accuracy.
3) We validate the practical usability of the converted roads through simulations in the Udacity self-driving car simulator using ADAS available from the literature. Our findings show how reliance on specific ADAS models impacts the interoperability of existing scenarios in SENSODAT, thereby calling for solutions to enhance their reusability.

The paper is organized as follows: Section II provides background information about lane-keeping ADAS and the SENSODAT benchmark. Section III overviews the OpenDRIVE format and the Catmull Rom spline. Section IV presents OPEN-CAT, that is, the conversion algorithm used for translating roads from the OpenDRIVE to the Catmull Rom Spline format. The results and experiments are discussed in Section V. Finally, Section VI provides concluding remarks.

II. BACKGROUND

A. Lane-keeping ADAS

We consider NHTSA [42] Level 2 ADAS, which relies on vision-based perception tasks using data captured by vehicle-mounted sensors. This paper focuses on ADAS which provides the *lane-keeping* functionality from human-labeled driving data. Lane-keeping is a critical component for the safe operation of ADS. Models such as NVIDIA's Dave-2 [43] learn to drive by identifying latent patterns in a training dataset composed of images captured during expert human driving. These models predict driving commands that replicate the behavior of a skilled driver. In its simplest form, a lane-keeping ADAS, such as Dave-2, can be represented as a function
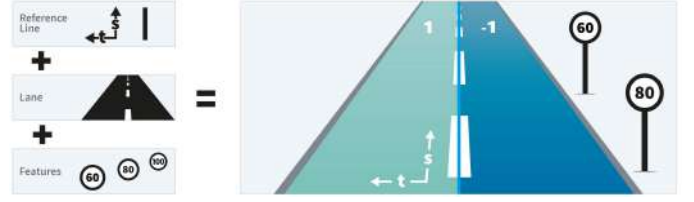

Figure 1: Elements of OpenDRIVE [45].

$f : \mathbb{R}^d \rightarrow [-25°, +25°]$. Here, d denotes the dimension of the input image $\boldsymbol{x} \in \mathbb{R}^d$ (e.g., for an image of size 140×320, $d = 44,800$ pixels), and the output is a scalar value y representing the predicted steering angle. This angle lies within the range $[-25°, +25°]$, where $-25°$ indicates maximum steering to the left, $+25°$ indicates maximum steering to the right, and $0°$ corresponds to no steering input.[1]

B. Benchmark

The SENSODAT dataset [33] comprises 32,580 test scenarios (i.e., roads) generated using existing test case generators. These scenarios encompass a wide range of road conditions, including curved roads, sharp turns, and complex layouts, enabling the testing of self-driving systems, such as lane-keeping models. The test scenarios were executed within the BeamNG simulator [38] using its AI driver autopilot,[2] a PID-based controller with global knowledge of the road topology. While SENSODAT mainly serves as a benchmark for sensor data, the authors highlight its potential for regression testing purposes, as trajectory data are also included. The dataset reports 19,926 successful test executions and 12,654 failures.

In this paper, we explore the interoperability of SENSODAT within the context of a different simulator (Udacity) and a learning-based driving model (Dave-2).

III. OVERVIEW OF OPENDRIVE FORMAT AND CATMULL ROM SPLINE

SENSODAT uses the OpenDRIVE format, which required us to convert the road topologies to the Catmull-Rom spline format used in Udacity. In this section, we provide a concise overview of the OpenDRIVE road format and Catmull-Rom spline, highlighting the key differences, followed by a detailed explanation of our conversion method in Section IV.

A. OpenDRIVE

ASAM OpenDRIVE [45] provides a comprehensive extensible markup language (XML)-based format (.xodr) for modeling networks in ADS simulators. An OpenDRIVE file contains data that describes the geometry of roads, lanes, and objects—such as road markings—as well as features along the roads, including signals as shown in Figure 1. The reference line, the backbone of the format, is constructed using concatenated clothoids (Euler spirals) [46], arcs, and polynomials.

---

[1]These values reflect the steering limits of an ADAS within the Udacity driving simulator [44].
[2]This conservative assumption is made due to the lack of specific details provided in the original paper.
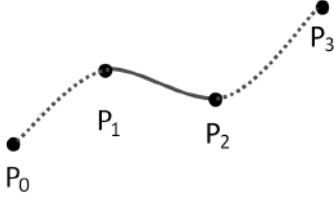
Figure 2: Catmull-Rom spline interpolation with four points.

This ensures smooth transitions in road curvature. In addition, lanes are specified by type, width, and position relative to the reference line and may be modified sectionally. Junctions are modeled by linking roads and lanes, allowing for complex traffic simulations [47].

SENSODAT does not include junctions, as the roads are treated individually rather than as a connected map: each road is identified by a unique road ID and processed separately. Additionally, SENSODAT does not exploit the flexibility of OpenDRIVE, and in this case, the richness of the format is mostly a source of complexity. Thus, researchers have proposed alternative road formats [48], like the Catmull-Rom spline. Unlike OpenDRIVE, it does not rely on a reference line and provides a simpler and more efficient road representation.

B. Catmull-Rom Spline

A Catmull-Rom spline [41] is a smooth curve (interpolation spline) widely used in computer graphics, animation, and path generation to create natural transitions between points. It passes directly through each control point (no approximation) while maintaining smoothness, meaning that there are no sharp angles or sudden changes in direction. One of its advantages is its localization control, which means that changing one point only affects the neighboring segments, not the entire curve, making it easy to tweak without disturbing the whole shape.

From a mathematical point of view, the spline uses cubic polynomials to calculate how the curve behaves between four consecutive points; the key idea here is that it creates a smooth transition between any two points while considering their neighbor for context. There is also a tension parameter that controls how tightly the curve bends around the points. Higher tension leads to sharper turns, while lower tension results in a more relaxed flowing curve [41].

Given a series of control points P_0, P_1, P_2, P_3 (Figure 2), the Catmull-Rom spline segment between P_1 and P_2 is computed using the following cubic Hermite interpolation formula:

$$C(t) = \frac{1}{2}\Big[(2P_1) + (-P_0 + P_2)t + (2P_0 - 5P_1 + 4P_2 - P_3)t^2$$
$$+ (-P_0 + 3P_1 - 3P_2 + P_3)t^3\Big]$$

(1)

where

- $C(t)$ is the position on the time. t.
- P_0, P_1, P_2, P_3 are control points.
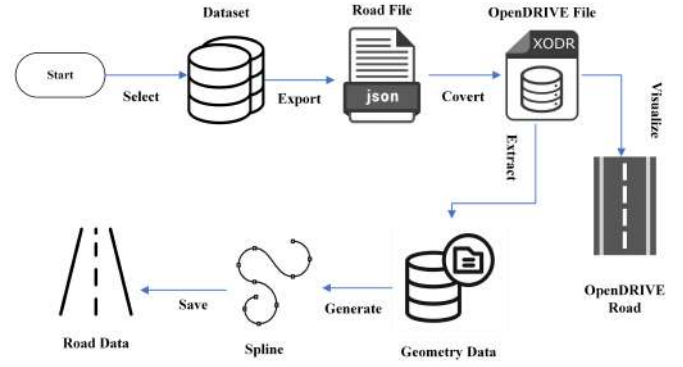- t ranges from 0 to 1, interpolating between P_1 and P_2.

Figure 3: Conversion process from OpenDRIVE format to Catmull-Rom spline.

This equation ensures that at $t = 0$, $C(t) = P_1$, the curve starts at the second control point, whereas at $t = 1$, $C(t) = P_2$, the curve ends at the third control point.

To create a complete spline, this process is repeated for each pair of points, connecting small segments smoothly. The advantages of this spline include its local control, smoothness, and computational efficiency. However, if the control points are unevenly spaced, the curve can overshoot. Also, while the curve is smooth, it does not always guarantee a perfect curvature at each point, which can be a limitation in precision-demanding applications like robotics. Despite these minor drawbacks, Catmull-Rom splines remain adequate for generating road topologies in a versatile manner [49].

IV. OPENCAT: FROM OPENDRIVE TO CATMULL ROM SPLINE

This section provides an overview of the road conversion process from OpenDRIVE to Catmull-Rom spline. The conversion process involves simplifying the detailed road model to focus on its geometry, specifically its shape and path. The diagram in Figure 3 illustrates the conversion process. First, we export the OpenDRIVE road field from the dataset scenario files in JSON format. These JSON files were then converted to OpenDRIVE (.xodr) files used by OPENCAT. After that, we extract the necessary geometry attributes. Then, we calculate the reference line (centerline) from the OpenDRIVE data, which involves averaging lane boundaries (right and left). The OpenDRIVE road network is built around a central reference line, which acts as the core element of every road. The reference line defines how the road bends and turns from the planView. All lanes and elevation profiles are linked to this reference line as shown in Figure 1.

Next, the specific points that define a spline known as control points are defined along this reference line to generate the Catmull-Rom spline, ensuring that they are spaced to accurately capture the road's curvature. Additionally, we integrate elevation data from the OpenDRIVE file by applying the elevation profile to the spline, although this step is challenging because splines typically do not directly account for such details. The whole process of data extraction and spline generation is explained next.

Algorithm 1 extract_road_geometry(*opendrive, lane_side*)

1: *road_data* ← []   ▷ Initialize a list to store road data
2: **for** each *road* in *opendrive* **do**
3:     *planView* ← *road*.planView
4:     **for** each *geometry* in *planView*.geometries **do**   ▷ Iterate through geometries of the road
5:        (x, y, s) ← *parse_geometry*(geometry)   ▷ Extract geometry attributes
6:        *elevation_width* ← *get_elevation_and_width*(road, s)
7:        *lane_offset* ← *compute_lane_offset*(road, *s*,lane_side)   ▷ lane_side is either left or right
8:        *adjusted_x* ← x + *lane_offset*.x   ▷ Adjust x-coordinate for lane
9:        *adjusted_y* ← y + *lane_offset*.y   ▷ Adjust y-coordinate for lane
10:        *append* (adjusted_x, adjusted_y, elevation_width.z, elevation_width.width) to *road_data*
11:     **end for**
12: **end for**
13: **return** *road_data*

Algorithm 2 generate_spline(*opendrive, lane_side*)

1: *right_lane_points* ← extract_road_geometry(*opendrive, right*)   ▷ Extract right lane points
2: *left_lane_points* ← extract_road_geometry(*opendrive, left*)   ▷ Extract left lane points
3: *min_points* ← 4   ▷ Minimum points for spline generation
4: **if** len (*right_lane_points*) **or** len (*left_lane_points*) < *min_points* **then**
5:     **return** *empty_list*
6: **end if**
7: *control_points* ← *compute_centerline*(right_lane_points, left_lane_points)   ▷ Compute centerline
8: *spline_generator* ← *compute_catmull_rom_spline*(control_points)   ▷ Generate spline
9: *spline_points* ← *spline_generator*.generate_spline()
10: **return** *control_points, spline_points*

A. Conversion Algorithm

Algorithm 1 is designed to process an OpenDRIVE file and extract the road geometry for a specified lane side (right, left), which will be used to generate a spline. For each road in an OpenDrive file, the algorithm iterates through its geometry sections to extract the attributes such as position (x, y), and starting position (s) by calling the auxiliary function *parse_geometry* (Line 5). Next, it calculates the elevation profile and width of the road at the current position *'s'* based on polynomial coefficients found in the OpenDRIVE data (Line 6). Then we compute the lane offset (Line 7), which is the total width of the lanes from the road center, depending on the specified lane side (left or right). Based on the *lane_side* parameter, the algorithm adjusts the x and y coordinates by the computed lane offset (Lines 8-9). This adjustment ensures that only the points from the desired lane side are extracted.

Finally, the extracted road data *(x, y, z, width)* are collected (line 10) and returned (line 13). Once the road geometry has been fully extracted, we proceed to generate the spline curve using *Algorithm 2*, which constructs a Catmull-Rom spline and interpolates points along the spline, ensuring that the road geometry is represented as a smooth curve. Details of the supporting functions used in this process are explained below.

1) Lateral and Elevation Profile

The *parse_geometry* function is designed to extract the basic geometry attributes of a road from the OpenDRIVE XML structure. It parses the geometry section of a road, which contains essential attributes such as *'x'* and *'y'* coordinates, *'length'* (the length of the geometry section), *'hdg'* (the heading or direction of the road at the starting point), and *'s'* (the position along the road where the geometry section begins). These values provide the fundamental data that describes the shape and orientation of the road section. This information is used to calculate the road geometry and lane offsets.

Although the data extracted through the parse_geometry function is sufficient for generating a flat 2D plane that represents the road's horizontal geometry, it does not capture the true 3D nature of the road terrain, such as slopes. To address this, we use the *get_elevation_and_width* function to extract both the elevation and the width of the road at a specified position *'s'*, ensuring a more accurate road representation. If the current road has an associated elevation profile, it iterates over the elevation segments and calculates the elevation for the given position (s) using polynomial coefficients. For each segment, it checks whether the current position *'s'* lies within that segment by comparing *'s'* with the starting point *'s_elev'* of the elevation and the length of the segment. If *'s'* falls within this range, the function calculates the elevation *'z'* at this position using a cubic polynomial formula:

$$z = a + b \cdot ds + c \cdot ds^2 + d \cdot ds^3 \tag{2}$$

Here *(a, b, c, d)* are polynomial coefficients, and *'ds'* is the difference between *'s'* and the start of the segment *(ds = s - s_elev)*. After calculating the elevation, the function extracts the road's width by locating the lanes within the *'laneSection'*. For each lane in the section, it calculates the width using similar polynomial coefficients and sums the widths to get the total road width. The calculated elevation and width are then returned, providing the necessary data to represent the physical characteristics of the road at that position.

2) Lane Offset

The *compute_lane_offset* function determines the lateral positioning (side-to-side) of the lanes on the road at a given

(a) Original OpenDRIVE road.   (b) Generated spline with original points.   (c) Road in Udacity simulator.
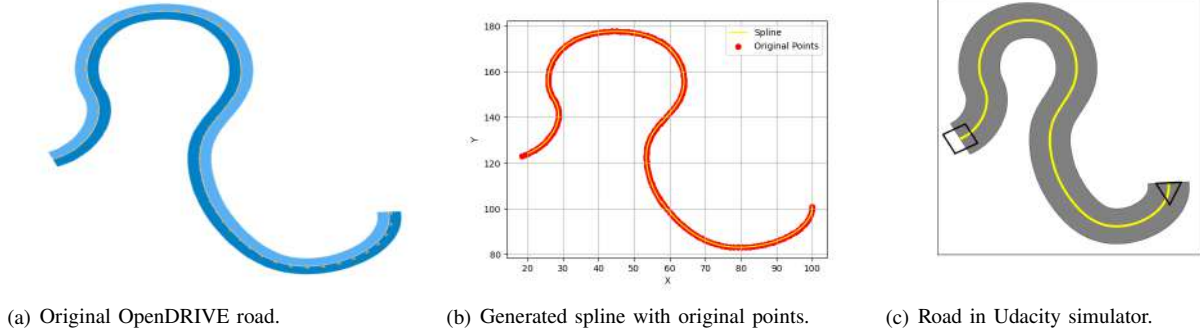
Figure 4: Comparative view of OpenDRIVE road, generated spline, and test case.

point (s) along the road geometry. It is essential for accurate road modeling, simulation, and visualization, especially when dealing with road curvature or varying lane widths.

First, we check the validity of each lane in the lane section on the road, skipping over the center lane *(lane_id = 0)*. For valid lanes, the width of each lane is calculated using polynomial coefficients, much like the elevation and width calculations earlier. The width of each lane is then added to the *'lane_offset'*, which accumulates the total offset of all valid lanes for the specified side. Once all lanes have been processed, the total lane offset is returned.

Given the previous information, we now have details on how to generate a spline. *Algorithm 2* generates a smooth spline (curve) representing the road's reference line (centerline). It begins by calling *Algorithm 1* to extract road geometry data for the specified lane side (*right* or *left*) from the OpenDRIVE file (Lines 1-2). These points represent the lane positions along the road. Next, the algorithm ensures there are enough points to generate a spline by checking if both the right and left lane data meet the minimum point requirement (Lines 3). If there are enough points, the algorithm proceeds by calculating the road's centerline by averaging the points of the right and left lanes to determine a set of control points that represent the middle of the road (Line 7). With these control points in hand, the algorithm initializes a *'CatmullRomSpline'* generator, which creates a smooth curve (spline) based on the control points (Line 8). To control the smoothness of the curve, the generator is configured with an *'alpha'* value of 0.5 and is set to produce one spline point per segment. Finally, the algorithm generates the spline points and returns both the control points and the generated spline points (Lines 9-10).

V. EXPERIMENTS AND RESULTS

To evaluate the performance of OPENCAT, we applied it to the publicly available OpenDRIVE scenario dataset called SENSODAT [33]. More in detail, we aim to answer the following research questions:

RQ₁ (Accuracy): How accurately does OPENCAT transform the OpenDRIVE road geometry into the Catmull-Rom spline representation?

RQ₂ (Comparison): How does the converted dataset perform relative to the original SENSODAT dataset in terms of the test pass/fail ratio?

A. RQ₁ (Accuracy)

We converted a total of 32,580 roads from the dataset, but for comparison, we visually present only a single example of conversion; other results can be verified at the publicly available GitHub repository [50]. Before converting the Open-DRIVE files into the spline, we verified the validity and usability of the data in the OpenDRIVE files by visualizing the road structure using the OpenDRIVE file viewers [51], [52]. Figure 4(a) presents an original road from the OpenDRIVE scenario visualized through the OpenDRIVE viewer [51], while Figure 4(b) presents the conversion result, where the generated spline (yellow) is overlaid with the original points from the OpenDRIVE file (red dots). We can see that the generated spline closely follows the original geometry, demonstrating the accuracy of the conversion. The visual results confirm that OPENCAT preserves the original geometric characteristics of the road while producing a smooth representation suitable for use in simulations.

In addition to visual validation, we also verify our results using the following metrics to measure the precision of the generated spline by comparing the spline points with the original points. The accuracy metric evaluates how closely the spline interpolation matches the original points by calculating the average Euclidean distance between the spline points and the original points. We then normalize this average distance by the maximum possible error, defined as the diagonal distance of the bounding box that encloses the original points. The resulting percentage indicates how accurate the spline interpolation is with respect to the original points:

$$\text{Accuracy} = \left(1 - \frac{\text{Average Distance}}{\text{Max Possible Error}}\right) \times 100 \quad (3)$$

R-squared (coefficient of determination) quantifies the goodness of fit of the spline interpolation compared to the original data. It measures how well the spline points fit the original

5

data. An R² value closer to 1 indicates that the spline fits the data well, while lower values signify a poor fit.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \qquad (4)$$

where y_i are the actual values. $\hat{y}_i$ are the predicted/interpolated values and $\bar{y}$ is the mean of the original values. The conversion achieved perfect accuracy (accuracy = 100%) and R-squared (R^2=1), confirming the correctness and quality of OpenCat.

B. RQ₂ (Comparison)

We implemented and validated our spline-based roads for the Udacity simulator [39] using the test case generation framework by Biagiola et al. [53]. The framework uses a DAVE-2 model as a system under test [43], which is an advanced driver assistance system for lane-keeping.

First, we initialize road scenarios by defining control points derived from our generated spline, similar to the test case representation in Biagiola et al. [53]. The control points represent critical aspects of road geometry from the OpenDRIVE file, such as lane boundaries, curves, and elevation changes. After interpolation, a road is considered valid if it meets certain criteria, which include: (1) the start and end points are distinct; (2) the road remains within a predefined square bounding box of 250 × 250 units; and (3) there are no intersections. Figure 4(c) presents an example of such a valid road generated by the Udacity test generator using our road data. We executed simulations on a MacBook Pro M2, 8 cores, 16 GB RAM, macOS Sequoia, and on an Intel Core i5, 12 cores, 16 threads, 4GB DDR6 RAM, NVIDIA GeForce RTX 3050 Ti, 16 GB, Windows 11.

Table I specifies the campaigns executed on either macOS or Windows, with (m) indicating macOS and (w) indicating Windows. This dual-system approach also helps further validate our methodology by testing the OpenCat's converted dataset performance on two different operating systems.

A test case is considered successful if the vehicle stays within the correct lane (right) up to the final road control point. In contrast, a test case fails if the car goes out of bounds (OOB), i.e., off the road [54], or if the road's endpoint overlaps with the starting point (i.e., road_818 from campaign_2_ambiegen), which prevents the simulator from starting, consequently failing the test case.

Each scenario began with the vehicle positioned at the starting control point, centered within the lane, and orientated to match the direction of the road, which helped establish a consistent starting condition across tests. As the vehicle drives along the paths defined by the spline, we monitored essential metrics to ensure realistic driving behavior, as outlined in previous work [53]. These metrics included lane-keeping accuracy, which checked the vehicle's lateral position to ensure it remained within lane boundaries, and OOB detection. This process allowed us to track safety violations [53].

Table I presents the simulation results from our simulation; we executed 32,580 tests in total. The results of our

Table I: Test pass/fail counts across campaigns.

Campaign	Pass	Fail	Total	Exe_Time h/m	OpenCat Pass%	SensoDat Pass%
Ambeigen Campaigns						
2_ambiegen(w)	963	10	973	5/19	98.97	58.17
3_ambiegen(w)	964	0	964	5/41	100	53.32
4_ambiegen(m)	956	9	965	5/50	99.06	51.30
5_ambiegen(m)	952	6	958	5/23	99.37	56.68
6_ambiegen(m)	951	8	959	5/44	99.16	54.12
7_ambiegen(m)	952	11	963	5/46	98.85	53.80
8_ambiegen(m)	941	11	952	5/46	98.84	54.31
9_ambiegen(m)	944	9	953	5/57	99.05	50.79
10_ambiegen(m)	964	7	971	5/38	99.27	53.88
11_ambiegen(m)	966	7	973	5/37	99.28	50.57
13_ambiegen(w)	944	10	954	5/34	98.95	53.85
14_ambiegen(m)	951	8	959	5/43	99.16	56.94
15_ambiegen(m)	939	13	952	5/33	98.63	54.20
Frenetic Campaigns						
2_frenetic(m)	920	8	928	5/38	99.13	60.56
3_frenetic(m)	944	10	954	5/36	98.95	63.34
4_frenetic(m)	956	8	964	5/44	99.17	65.06
5_frenetic(m)	935	10	945	5/39	98.94	66.67
6_frenetic(w)	854	90	944	5/33	90.46	65.93
7_frenetic(m)	958	9	967	5/53	99.06	64.33
8_frenetic(w)	859	93	952	5/37	94.01	63.87
9_frenetic(m)	959	5	964	5/51	99.48	64.20
11_frenetic(w)	828	28	866	5/17	96.76	63.51
12_frenetic(w)	928	28	956	5/38	97.07	63.07
13_frenetic(w)	925	34	959	5/38	96.45	67.06
14_frenetic(m)	857	9	866	5/10	98.96	68.59
15_frenetic(m)	860	10	870	5/5	98.87	61.84
Frenetic_v Campaigns						
2_frenetic_v(w)	943	1	944	5/51	99.90	67.63
4_frenetic_v(m)	524	1	525	3/16	99.80	65.52
5_frenetic_v(w)	940	0	940	5/49	100	67.45
6_frenetic_v(w)	763	1	764	4/42	99.86	66.58
7_frenetic_v(w)	47	0	47	0/18	100	55.32
11_frenetic_v(w)	949	4	953	5/58	99.58	65.27
12_frenetic_v(w)	888	54	942	5/53	94.26	66.24
13_frenetic_v(m)	946	5	951	6/1	99.47	70.97
14_frenetic_v(m)	928	6	934	5/44	99.35	72.93
15_frenetic_v(w)	925	24	949	5/58	97.47	64.18

experiments reveal a significant improvement in the ratio of passing tests across all test campaigns compared to the original SensoDat [33] based on the out_of_bound (OOB) metric. The table employs a color scheme to distinguish the comparison between roads generated through OpenCat's translation and those provided by SensoDat. The green color indicates campaigns where OpenCat outperformed SensoDat with a percentage difference greater than 30%, the yellow color represents campaigns with a difference of 25% to 30%, and the red color indicates cases where the difference was less than 25%. Unlike the original dataset, where pass rates varied considerably—especially in challenging road structures such as Frenetic campaigns—our results show consistently high pass rates, with 32,035 passing tests and 545 failing tests, resulting in an overall pass ratio of 98%, compared to a 61% pass ratio of the original SensoDat data set, which recorded 19,926 passing tests and 12,654 failing tests.

In particular, while the Frenetic campaigns in SensoDat displayed several failure rates, our results show that an independent ADAS model achieves substantial success rates, even the most challenging campaigns achieving over 95% pass rates, except for a few specific cases, such as campaign_6_frenetic and campaign_8_frenetic. Ambiegen campaigns [25], which performed very low in the original dataset, performed exceptionally well in our results, with pass rates consistently

exceeding 98%, compared to the range of 50% to 58%, while macOS and Windows executions delivered similar results, highlighting the robustness of our evaluation. Similarly, FreneticV campaigns, which generally showed strong pass rates in SensoDat, displayed near-ideal results in our simulations. By including execution time as a metric, we provide insight into the computational efficiency of different campaigns, offering valuable information to optimize resource allocation in future simulations.

Our findings demonstrate significant improvements in test pass rates compared to SensoDat and reveal an important limitation in current ADAS benchmarks. While the converted geometry is identical to the original, as established in RQ$_1$, the significant difference in test pass/fail results can be attributed to the interaction between the representation format, the ADAS under test (e.g., Dave-2), and the simulator environment.

Future work will investigate the specific factors contributing to the observed results. Moreover, benchmarks like SensoDat are coupled to the specific ADAS model with which they were created, limiting their utility in evaluating alternative ADAS systems. The ADAS under test, independent of SensoDat, successfully navigates most roads with minimal issues, highlighting the need for architecture-agnostic benchmarks for regression testing purposes. We propose the development of ADS-independent benchmarks that enable meaningful comparisons across diverse ADAS models and systems. This represents an open research problem in ADAS testing and a critical area for future investigation.

VI. Conclusions

In this study, we introduced OpenCat, a novel tool designed to convert the OpenDRIVE road format into Catmull-Rom spline representations, helping to bridge the gap in road format interoperability for autonomous driving applications.

Our converter OpenCat simplifies the complexities of generating precise road maps, making them adaptable for simulation to test lane-keeping ADAS models. By applying it to the SensoDat dataset, we demonstrated its effectiveness in accurately translating OpenDRIVE road geometry into realistic and smooth road scenarios. The integration of converted roads into the Udacity self-driving car simulator further validated OpenCat's capacity to generate reliable, simulation-ready roads suitable for diverse autonomous driving test environments. Our study demonstrates that ADAS under test, independent of SensoDat, successfully navigates a significant portion of the roads (98%) without any failure. This indicates that the effectiveness of SensoDat as a regression testing benchmark may be limited and we propose developing architecture-agnostic benchmarks as a crucial step toward advancing ADAS testing and evaluation.

References

[1] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y.-F. Li, L. Ma, Y. Xue, and Y. Liu, "A Survey on Automated Driving System Testing: Landscapes and Trends," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 5, Jul. 2023.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[3] Y. Li and L. Xu, "Panoptic perception for autonomous driving: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2408.15388

[4] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[5] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing Machine Learning based Systems: A Systematic Mapping," *Empirical Software Engineering*, 2020.

[6] G. Elghazaly, R. Frank, S. Harvey, and S. Safko, "High-definition maps: Comprehensive survey, challenges, and future perspectives," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 527–550, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259915452

[7] A. Banafa, "Technical challenges in autonomous vehicle development," https://www.linkedin.com/pulse/technical-challenges-autonomous-vehicle-development-banafa-ypfhc/, 2024, accessed: 2024-11-03.

[8] A. Stocco, P. J. Nunes, M. d'Amorim, and P. Tonella, "ThirdEye: Attention Maps for Safe Autonomous Driving Systems," in *Proceedings of 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. IEEE/ACM, 2022.

[9] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of ACM 42nd International Conference on Software Engineering*, ser. ICSE '20. ACM, 2020, p. 12 pages.

[10] A. Stocco, B. Pulfer, and P. Tonella, "Model vs system level testing of autonomous driving systems: a replication and extension study," *Empirical Software Engineering*, vol. 28, no. 3, p. 73, May 2023.

[11] A. Stocco and P. Tonella, "Confidence-driven weighted retraining for predicting safety-critical failures in autonomous driving systems," *Journal of Software: Evolution and Process*, 2021. [Online]. Available: https://doi.org/10.1002/smr.2386

[12] S. C. Lambertenghi and A. Stocco, "Assessing quality metrics for neural reality gap input mitigation in autonomous driving testing," in *Proceedings of the IEEE 17th International Conference on Software Testing, Verification and Validation*, ser. ICST '24. IEEE, 2024, p. 12 pages.

[13] S. C. Lambertenghi, H. Leonhard, and A. Stocco, "Benchmarking image perturbations for testing automated driving assistance systems," in *Proceedings of the IEEE 18th IEEE International Conference on Software Testing, Verification and Validation*, ser. ICST '25. IEEE, 2025, p. 12 pages.

[14] A. Stocco, B. Pulfer, and P. Tonella, "Mind the Gap! A Study on the Transferability of Virtual Versus Physical-World Testing of Autonomous Driving Systems," *IEEE Transactions on Software Engineering*, vol. 49, no. 04, pp. 1928–1940, apr 2023.

[15] B. Kim, Y. Kashiba, S. Dai, and S. Shiraishi, "Testing autonomous vehicle software in the virtual prototyping environment," *IEEE Embedded Systems Letters*, vol. 9, no. 1, pp. 5–8, 2017.

[16] M. R. Zofka, S. Klemm, F. Kuhnt, T. Schamm, and J. M. Z"ollner, "Testing and validating high level components for automated driving: Simulation framework for traffic scenarios," in *Proceedins of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 144–150.

[17] N. Neelofar and A. Aleti, "Identifying and explaining safety-critical scenarios for autonomous vehicles via key features," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 4, Apr. 2024.

[18] ——, "Towards Reliable AI: Adequacy Metrics for Ensuring the Quality of System-level Testing of Autonomous Vehicles," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. ACM, 2024.

[19] V. Crespo-Rodriguez, Neelofar, and A. Aleti, "PAFOT: A Position-Based Approach for Finding Optimal Tests of Autonomous Vehicles," in *Proceedings of the ACM/IEEE 5th International Conference on Automation of Software Test (AST 2024)*, ser. AST '24. ACM, 2024, p. 159–170.

[20] C. Lu, S. Ali, and T. Yue, "Epitester: Testing autonomous vehicles with epigenetic algorithm and attention mechanism," *IEEE Transactions on Software Engineering*, pp. 1–19, 2024.

[21] C. Lu, T. Yue, M. Zhang, and S. Ali, "DeepQTest: Testing Autonomous Driving Systems with Reinforcement Learning and Real-world Weather Data," 2023. [Online]. Available: https://arxiv.org/abs/2310.05170

[22] Q. Pan, T. Wang, P. Arcaini, T. Yue, and S. Ali, "Safety assessment of vehicle characteristics variations in autonomous driving systems," 2023. [Online]. Available: https://arxiv.org/abs/2311.14461

[23] F. Klück, Y. Li, J. Tao, and F. Wotawa, "An empirical comparison of combinatorial testing and search-based testing in the context of automated and autonomous driving systems," *Information and Software Technology*, vol. 160, p. 107225, 2023.

[24] F. Klück, D. Sumann, and F. Wotawa, "Utilizing genetic algorithms for generating critical scenarios for testing autonomous driving functions," in *2024 IEEE International Conference on Artificial Intelligence Testing (AITest)*, 2024, pp. 73–80.

[25] D. Humeniuk, F. Khomh, and G. Antoniol, "Ambiegen: A search-based framework for autonomous systems testingimage 1," *Science of Computer Programming*, vol. 230, p. 102990, 2023.

[26] J. Wu, C. Lu, A. Arrieta, T. Yue, and S. Ali, "Reality bites: Assessing the realism of driving scenarios with large language models," in *Proceedings of the IEEE/ACM 2024 First International Conference on AI Foundation Models and Software Engineering (FORGE)*. ACM, 2024, p. 40–51.

[27] P. Arcaini and A. Cetinkaya, "Crag – a combinatorial testing-based generator of road geometries for ads testing," *Science of Computer Programming*, vol. 238, p. 103171, 2024.

[28] T. Laurent, S. Klikovits, P. Arcaini, F. Ishikawa, and A. Ventresque, "Parameter coverage for testing of autonomous driving systems under uncertainty," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 3, Apr. 2023.

[29] F. Khan, H. Anwar, and D. Pfahl, "Simulation-based safety testing of automated driving systems," in *Product-Focused Software Process Improvement*. Springer, 2024, pp. 133–138.

[30] ——, "A process for scenario prioritization and selection in simulation-based safety testing of automated driving systems," in *Product-Focused Software Process Improvement*. Springer, 2024, pp. 89–99.

[31] L. Baresi, D. Y. X. Hu, A. Stocco, and P. Tonella, "Efficient domain augmentation for autonomous driving testing using diffusion models," in *Proceedings of the IEEE 47th International Conference on Software Engineering*, ser. ICSE '25. IEEE, 2025.

[32] Q. Ali, O. Riganelli, and L. Mariani, "Testing in the evolving world of dl systems: Insights from python github projects," in *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*, 2024, pp. 25–35.

[33] C. Birchler, C. Rohrbach, T. Kehrer, and S. Panichella, "Sensodat: Simulation-based sensor dataset of self-driving cars," in *Proceedings of the 21st International Conference on Mining Software Repositories*, ser. MSR '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 510–514. [Online]. Available: https://doi.org/10.1145/3643991.3644891

[34] C. Lu, T. Yue, and S. Ali, "Deepscenario: An open driving scenario dataset for autonomous driving system testing," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, 2023, pp. 52–56.

[35] J. Dai, B. Gao, M. Luo, Z. Huang, Z. Li, Y. Zhang, and M. Yang, "Sctrans: Constructing a large public scenario dataset for simulation testing of autonomous driving systems," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3623350

[36] M. Borg, R. B. Abdessalem, S. Nejati, F.-X. Jegeden, and D. Shin, "Digital twins are not monozygotic–cross-replicating adas testing in two industry-grade automotive simulators," in *ICST '21*. IEEE, 2021.

[37] M. H. Amini, S. Naseri, and S. Nejati, "Evaluating the impact of flaky simulators on testing autonomous driving systems," *Empirical Softw. Engg.*, vol. 29, no. 2, feb 2024.

[38] BeamNG.tech, "BeamNG GmbH," https://beamng.tech/, 2024, online; accessed 2024-11-19.

[39] Udacity, "Self-driving car simulator," 2023, accessed: 2024-11-04. [Online]. Available: https://github.com/udacity/self-driving-car-sim

[40] M. Dupuis, M. Strobl, and H. Grezlikowski, "OpenDRIVE 2010 and beyond – status and future of the de facto standard for the description of road networks," in *Proceeding of the Driving Simulation Conference Europe*, 2010, pp. 231–242.

[41] E. E. Catmull and R. Rom, "A class of local interpolating splines," *Computer Aided Geometric Design*, pp. 317–326, 1974. [Online]. Available: https://api.semanticscholar.org/CorpusID:118383557

[42] U. D. of Transportation, "A framework for automated driving system testable cases and scenarios," https://rosap.ntl.bts.gov/view/dot/38824/dot_38824_DS1.pdf, 2018.

[43] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars." *CoRR*, vol. abs/1604.07316, 2016.

[44] Udacity, "Udacity self-driving car simulator," https://github.com/udacity/self-driving-car-sim, 2021, accessed: [2024-01-15].

[45] Association for Standardization of Automation and Measuring Systems (ASAM), "ASAM OpenDRIVE®," https://www.asam.net/standards/detail/opendrive/, 2023, accessed: 2024-11-04.

[46] K. Baass, "The use of clothoid templates in highway design," in *Transportation Forum*, vol. 1, 1984, pp. 47–52.

[47] B. Schwab and T. H. Kolbe, "Validation of parametric opendrive road space models," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. X-4/W2-2022, pp. 257–264, 2022. [Online]. Available: https://isprs-annals.copernicus.org/articles/X-4-W2-2022/257/2022/

[48] S. Klikovits, V. Riccio, E. Castellano, A. Cetinkaya, A. Gambi, and P. Arcaini, "Does road diversity really matter in testing automated driving systems? – a registered report," 2022. [Online]. Available: https://arxiv.org/abs/2209.05947

[49] S. Documentation, "Catmull-rom splines," 2024, accessed: 2024-11-19. [Online]. Available: https://splines.readthedocs.io/en/latest/euclidean/catmull-rom.html

[50] Q. Ali, "Opencat," 2024, accessed: 2024-11-19. [Online]. Available: https://github.com/lakhanqurban/OpenCat

[51] M. Pagel *et al.*, "libopendrive," 2024, accessed: 2024-11-04. [Online]. Available: https://github.com/pageldev/libOpenDRIVE

[52] S. Pagel, "Opendrive viewer," https://sebastian-pagel.net/odrviewer/, accessed: 2024-11-04.

[53] M. Biagiola, A. Stocco, V. Riccio, and P. Tonella, "Two is better than one: digital siblings to improve autonomous driving testing," *Empirical Software Engineering*, vol. 29, p. 72, 2024. [Online]. Available: https://doi.org/10.1007/s10664-024-10458-4

[54] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the ACM 28th SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 318–328.